Optimization and Simulation Winter 2021

Laboratory 6

Optimization Exercises
20.04.2021 & 27.04.2021

Melvin Wong

Transport and Mobility Laboratory TRANSP-OR

École Polytechnique Fédérale de Lausanne EPFL

Goals

- Understand the limitation of full enumeration
- Understand and apply optimization algorithms:
  - Greedy algorithm
  - Local search
  - Variable neighborhood search
  - Simulated annealing

Lab materials

- You will use the following Python libraries in this exercise:
  - `numpy, plotly`

Install the libraries using pip:

```
> pip install numpy plotly
```

Or if you're using Anaconda:

```
> conda install numpy
> conda install –c plotly plotly
```

You can use other plotting libraries yoo.

Overview

Travelling salesman problem

Implementation functions:

- Full enumeration
- Greedy algorithm
- Local search
- Variable neighborhood search
- Simulated annealing

Optimization Exercise 1 – Travelling Salesman Problem

Travelling Salesman Problem

A salesman must visit n cities

- He starts and ends the trip at her home city
- Assume cost of travel to be total trip length

What sequence of cities minimizes the travel cost?

Travelling Salesman Problem

Cities are consecutively numbered: $1, 2, \ldots, n$

We encode solutions as $x = (x_1, x_2, \ldots, x_n, x_1)$where

- x_1 is the index of the home city
- x_i is the index of i^{th} city visited along the way
- x_n is the last city visited before returning home
- Every city must be visited exactly once

Lab exercises – implementation of optimization algorithms

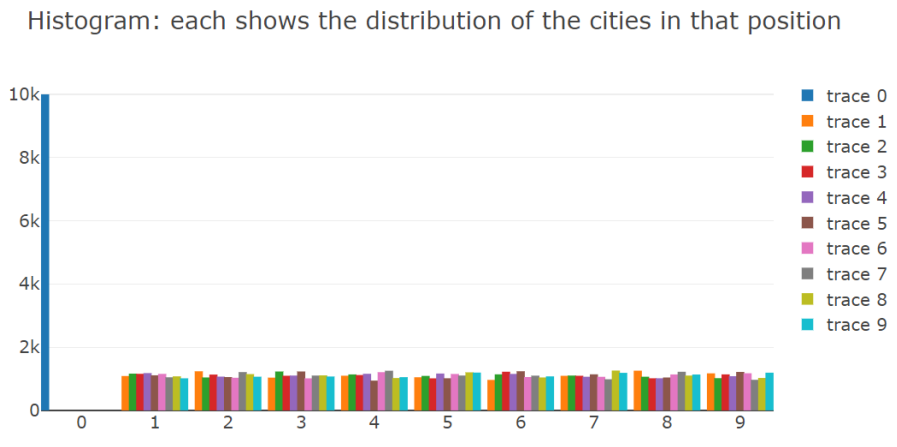
Core functionality

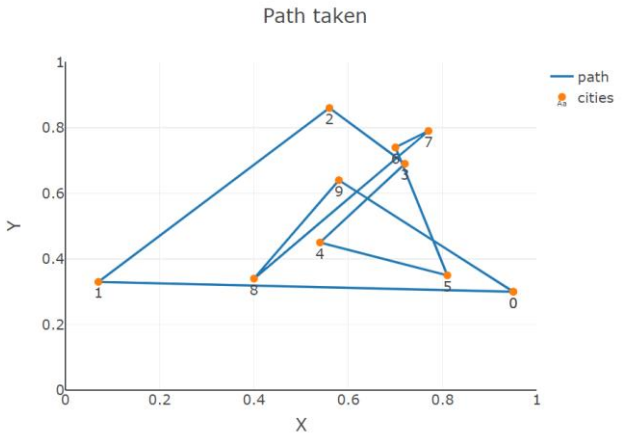
Function to implement

```
def simulateCities(n_cities, seed)
```

Test the function

```
def drawSalesman(path, cities)
```





Implementation functions

Objective Function

Function to implement

```
evaluate(path, cities)
```

Calculate the total distance travelled

Test the function

```
distance = evaluate(…)
```

Show that the distance traveled is accurate

6.1 Full enumeration

Exercise

1. Function to implement

```
generateNewCitySeq_fe()
FullEnumeration()
```

2. Test the function

```
OptimizationTSPTest()
```

Calculate the computational time limitations of the full enumeration.

What is the maximum problem size (number of cities) that you could solve with this approach?

6.2 Greedy algorithm

Greedy algorithm

In class Exercise

Function to implement

```
generateNewCitySeq_gs()
GreedySearch()
```

Test the function

```
OptimizationTSPTest()
```

6.3 Local search

Exercise

Function to implement

```
GenerateNewCitySequence_ls()
LocalSearch()
```

Test the function

```
OptimizationTSPTest()
```

6.4 Variable neighborhood search

Exercise

Function to implement

```
VNS()
```

Test the function

```
OptimizationTSPTest()
```

6.5 Simulated annealing

Exercise

Function to implement

```
generateNewCitySequence_sa()
temperature()
SimulatedAnnealing()
```

Test the function

```
OptimizationTSPTest()
```

- Investigate the effect of different parameters

IMPORTANT!

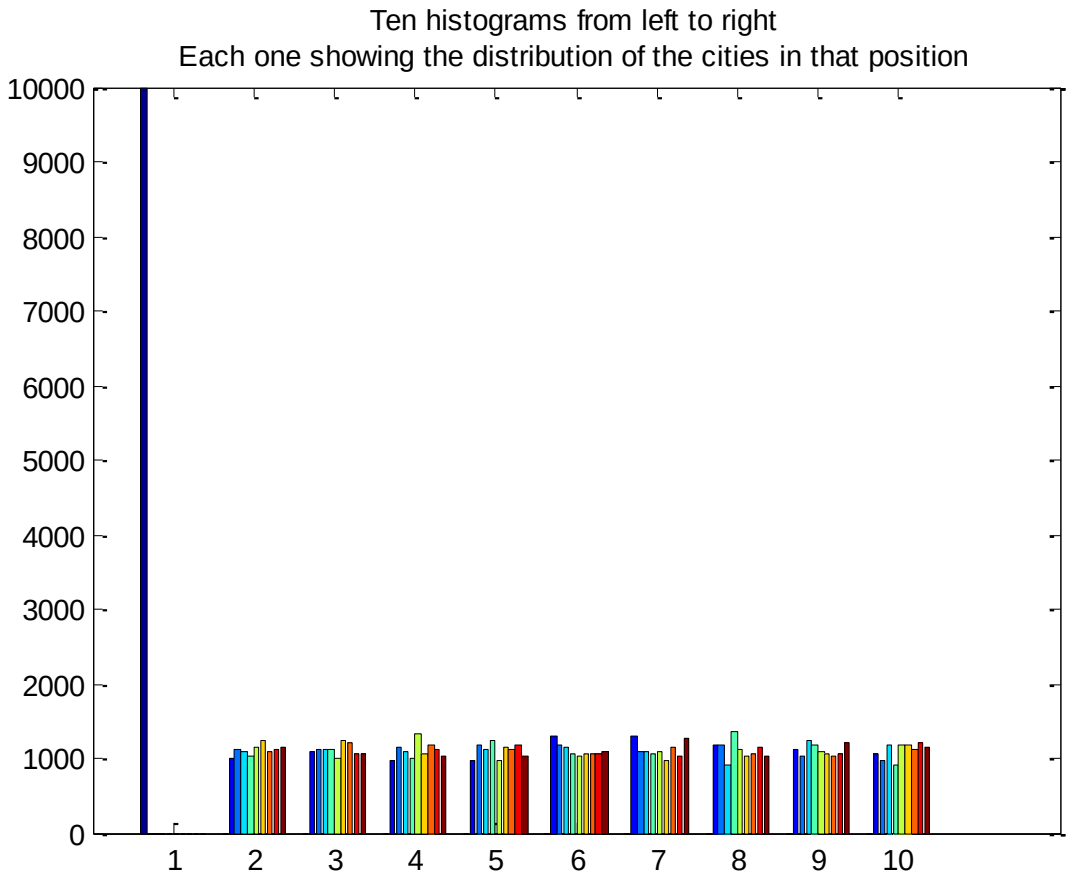
The suggested framework is only a **suggestion**!

Feel free to organize the code in the manner you find the most appropriate!

Sample results

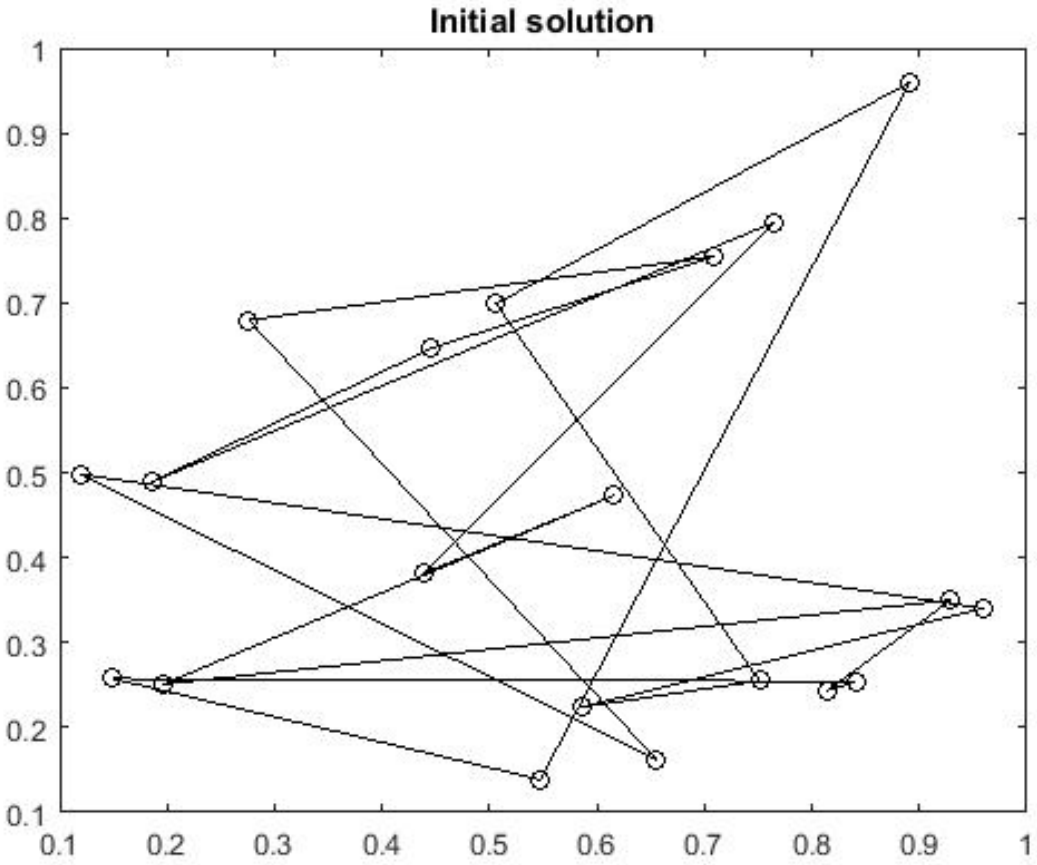
Sample results

`RandomizeCitySequence`



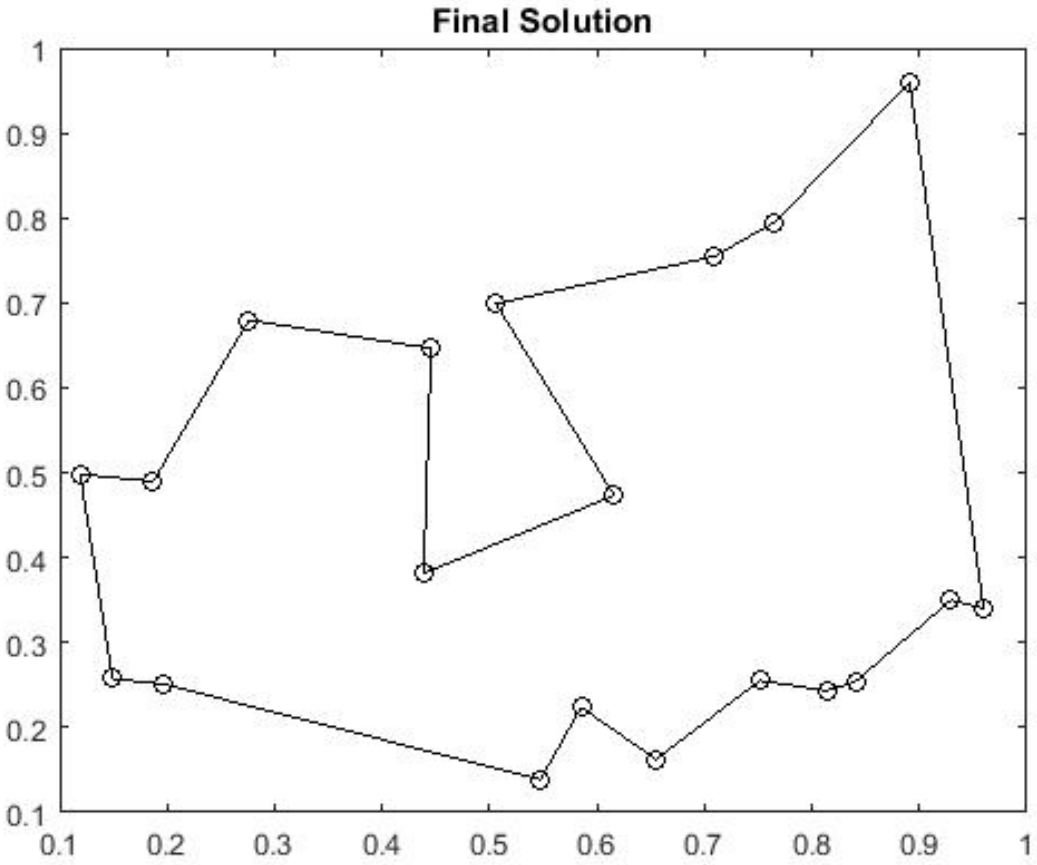
Sample results

An initial solution



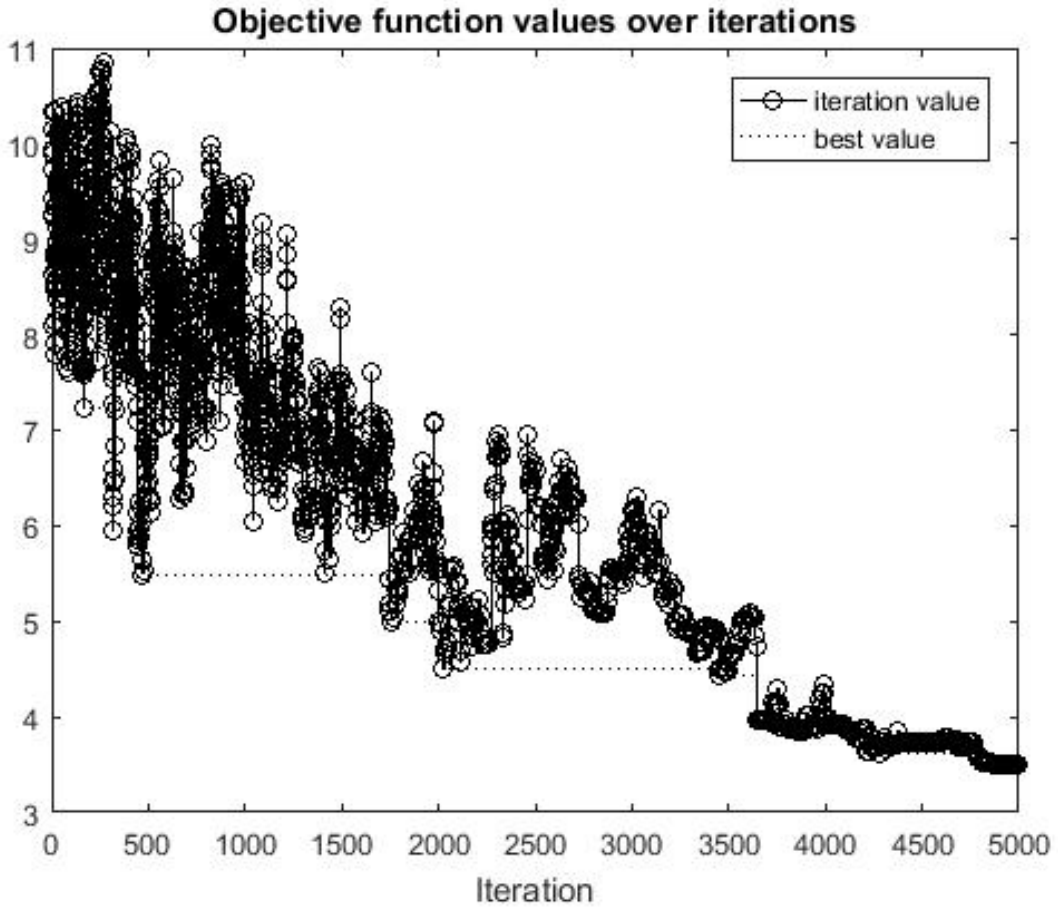
Sample results

A final solution



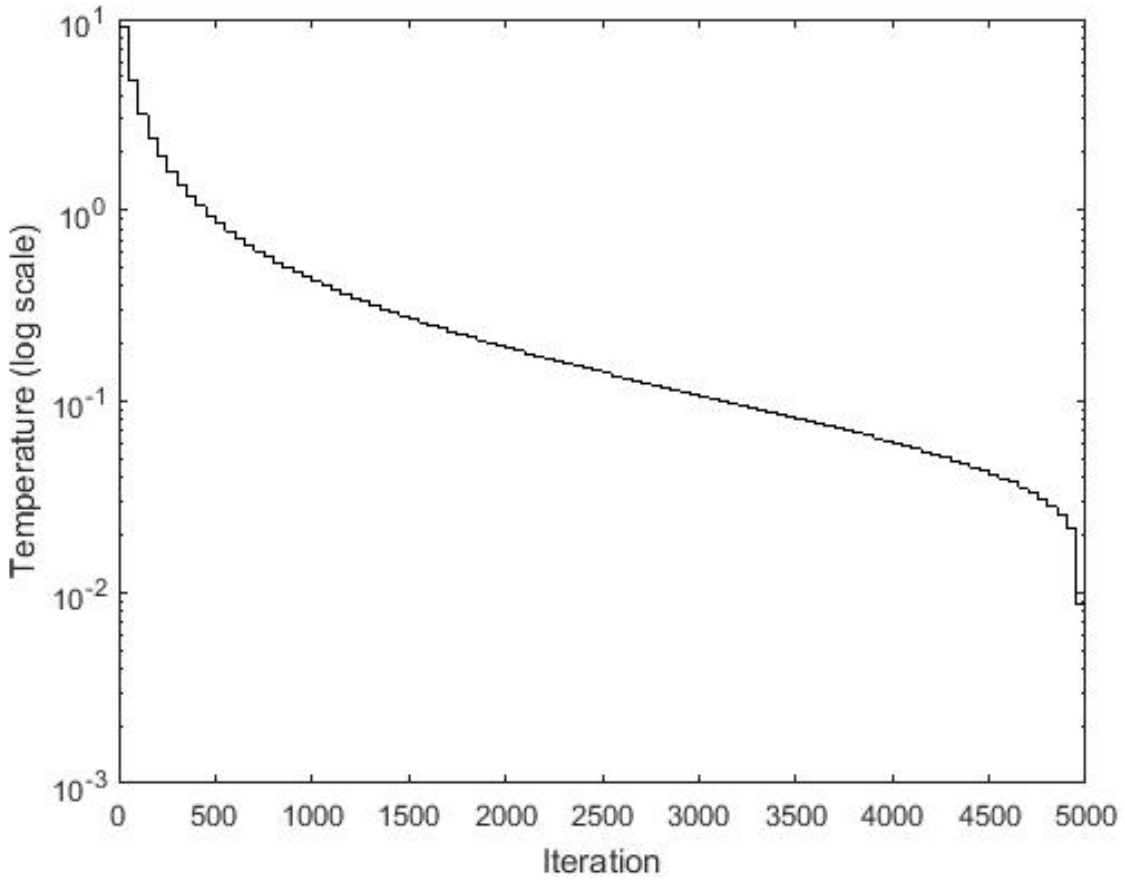
Sample results

Simulated Annealing



Sample results

Simulated Annealing



Optimization Exercise 2 – Knapsack problem

Knapsack Problem

- A salesman is planning his visit to the market and wants to select items he will try to sell.
- Each item i is characterized by its price p_i and weight w_i, which are given on the next slide
- There are 60 items in total.
- Logically, he would like to take as many items as possible, but his van has a weight limitation of max. 150 kilograms.

What is the set of items which will maximize the profit?

Knapsack Problem

- Price p_i and weight w_i for each item i:

i	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
p_i	80	31	48	17	27	84	34	39	46	58	23	67	62	79	38	44	31	50	72	71
w_i	84	27	47	22	21	96	42	46	54	53	32	78	64	82	33	49	28	56	77	69

i	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40
p_i	73	32	49	21	25	91	36	19	8	74	56	41	62	89	48	54	11	41	52	54
w_i	79	29	51	19	23	94	39	16	8	69	61	38	59	92	43	59	18	44	58	67

i	41	42	43	44	45	46	47	48	49	50	51	52	53	54	55	56	57	58	59	60
p_i	9	21	58	26	57	65	54	48	36	49	34	75	59	86	24	51	41	63	72	45
w_i	14	26	57	33	51	72	62	52	44	55	43	83	70	93	23	56	40	66	75	49

Tasks

- Try to solve the problem with the full enumeration
- Test the following optimization algorithms:
  - Variable neighborhood search (VNS), by implementing at least 3 neighborhood structures, and
  - Simulated annealing (SA)
- Use the full enumeration solution as a benchmark, if obtained.

Problem Encoding

- Let us label each item with an integer: $1, 2, \ldots, n$
- We encode solutions as $x = (x_1\ x_2\ \ldots\ x_i \ldots\ x_n)$ where
  - x_i is the 0-1 variable denoting if the item i is selected into the knapsack

Presentation of results

- Average number of appearances of each item in the knapsack
- Solutions achieved with all three algorithms
- Trend of the objective function value and temperatures in the case of SA

Deliverables

1. Python notebook file with your implementations and results
    1. **Travelling salesman problem**
    2. **Knapsack problem**

Submit as .ipynb file:

- Code
- Visualizations/Results

Also include other .py files that you used

Zip your files into one package with your project

Send it to the TAs by **01.06.2021 12 PM**